Procedure: ***Estimating Critical Resources***	
Issue Date: January 9, 2001	**Procedure ID:** ***P-PE-200***
Supersedes: May 5, 2000	**Rev/Change 2.1**

1. **Purpose:** To define the resources that are critical to the success of the project.

2. **Applicability:** This procedure is applicable to all government and contractor personnel assigned to ATISD.

3. **Responsibility:** Project Manager

4. **Support:** Engineers

5. **Invoked By:**

Define the Approach-	P-PE-010
Monitor and Control-	P-PM-150

6. **Inputs:**

   Performance Requirements
   System Architecture
   Top Level Design
   Operational Concept

7. **Outputs:**

Critical Computer Resources Worksheet	S-PE-270
Critical Computer Resources Tracking Form	S-PE-272

8. **Procedures Invoked:** N/A

9. **External Procedures Referenced:** N/A

10. **Procedure Steps:**

   a) The Chief Engineer identifies the system performance requirements and assures a complete understanding of those requirements.
   b) The Chief Engineer, with assistance from the other Engineers, analyzes the performance requirements to determine the software functions that must be executed to meet the requirement based on the system's architecture and top-level design.
   c) The Chief Engineer identifies the Critical Resources for the project. Throughput, CPU usage, RAM usage, Disk usage, and I/O required are the minimum resources to be addressed. The Chief Engineer records these results on the Critical Computer Resources Worksheet (S-PE-270). The remaining steps are performed only for the Critical Resources identified.

d) The Chief Engineer, with the assistance of the other Engineers, uses the system architecture, top-level design, hardware parameters and one or more operational scenarios to estimate the expected amount of usage for each critical resource. They prepare these estimates using analytical studies, simulations or other measurements (see Notes).
e) The Chief Engineer compares the estimated usage against the available (or allocated) capacity of the resource. If the capacity is adequate, no action is required. If the capacity is inadequate, the Chief Engineer reviews the requirements, revises the architecture and/or the top-level design, and/or hardware capacity and/or performance. (Note that early in the project, only rough estimates of the resource usage, and of the platform's performance can be made. Thus the performance of critical resources is periodically determined and tracked throughout the project.)
f) Once the design is considered feasible, the Chief Engineer sets the threshold for the critical resources. (These are called the allocated amounts.) The allocated amounts are based on parameters such as: record type, size, and number; number of users; number of I/O ports, etc., as well as operational scenarios. (All of these parameters should be documented in a Technical Note!)
g) The Chief Engineer records the allocated amounts, and initial estimates on worksheets S-PE-270 and S-PE-272. All of the assumed parameters (mentioned in step f) and any other assumed conditions associated with the performance measurement must be documented in a Technical Note, which is referenced on worksheet S-PE-272.
h) The Chief Engineer delivers these items to the Contractor Project Manager.
i) The Contractor Project Manager places the worksheet and forms in the Project Records.

11. Notes:

a) Here are some ideas for estimating and measuring computer resource utilization.

<u>Disk (Mass) Storage</u>

Estimate:

Physical Storage = ? (Record Size)(1+E)
files &
tables

+ ? (size(SLOC))(SLOC? Bytes)(1+E)Code
Software
(custom, OS & COSTS [RDBMS])

where

E = Storage overhead Factor (due to parity, CRC, etc.)

For size (SLOC) you can also use the measured size for the vendor's previous version, times (1+G) where G is the Estimated Growth.

Measurement:

Load actual code and data (if available) or "stubs" otherwise. Use the operating system's file management utilities to measure the storage used.

ROM, BIOS storage (For Programs and "Hard" Data)

Use the second term ("software") for disk storage above.

CPU Utilization

Background: CPU utilization is coupled to the architecture of the processor chip (e.g., cache, internal bus speed), the storage hierarchy (cache? RAM? disk? tape/CD ROM), and the I/O (memory access) bandwidth, as well as the partitioning of the processing functions and data, and the execution profile of the software. Given an architecture, platform design and software design (the Top-level design), the system designer usually performs trade studies to evaluate alternate partitioning, hardware capacities (e.g., buffer size and I/O rates), storage management protocols (e.g., write through caching, swapping and overlay algorithms) to "tune" the design. The fact that modern operating systems employ sophisticated (and proprietary) algorithms to improve performance makes CPU utilization very difficult to estimate. (For example, consider the algorithms for multiprogramming in multiprocessor configurations.)

Estimate:

Initially, these studies can be made with simple linear models. Later, simulation tools, available from the platform vendor may be used. For some systems, custom simulators may have to be built. A simple linear model is:

Total Instruction
Execution Rate = ? (# instructions/invocation)(# invocations/second)(1+G)
Major Functions

where:

Major Functions are the most frequently executed sections of code. (See below.)

# instructions - the number of machine instructions executed for an invocation of a particular major function

G - Process Management Overhead.

The Major Functions can be decomposed by their main loops, sorts & searches (see Note 1). If a mix of major functions is executed, the sum can be over the average mix of simultaneously executing functions or over a processing cycle (or "frame"). Alternately, if several sequences of functions, or a group of frames, executes, then the summation could be extended to use a weighted average of the number of invocations, e.g., ? $f_i N_{i.}$ Note that simple spreadsheets can be used to record instruction counts, invocation frequencies, weights, etc. (These are called "scoping models".) We recommend handling the estimates in this way for simple systems. (Complex systems will require use of queuing models (very difficult) or, more likely, custom simulations of the resource usage.)

Measurement:

Plan Ahead! Include “hooks” in the design to allow the injection of known workloads, collection of key performance parameters, and control of process initialization, states and sequencing. Possibly the operating system vendor can provide diagnostic/measurement tools as well.

<u>RAM Storage</u>

RAM storage is dependent on overlay schemes (disk swapping or “virtual memory”), as well as on the total size of the executable software and its associated dynamic data. The Disk Storage algorithm provides a starting point. Note that some functions are placed in ROM (e.g., BIOS functions).

Measurement:

Use the operating system’s memory management/viewing utilities.

<u>I/O Rates</u>

This is coupled to the process and memory management algorithms used by the platform (virtual machine), as well as the software’s design and operating profile. (See CPU Utilization and RAM Storage above.)

Estimate:

$$\text{I/O Rate} = \underset{\substack{\text{All types of} \\ \text{Messages \& Transactions} \\ \text{(In plus Out)}}}{?} (\text{\# Bytes/Message})(\text{\# Messages/Second})(1+F)$$

where F = Communications Protocol Overhead

Measurement:

Insert “hooks” to inject and measure the rates under controlled conditions. Possibly the vendor of the operating system/platform can provide diagnostic tools to measure I/O rates as well.

<u>Possible Refinements</u>

All of the estimates described above can be refined by using the PERT technique. For each estimated quantity, the estimators provide the lowest possible, most likely, and highest possible values (denoted by L, M, H, respectively). For a single estimated quantity, x, the PERT technique gives:

$$<x> = (L+4M+H)/6, \quad ?_{x} = (H - L)/6$$

Because we have the product of two estimated quantities (e.g., size and rate of occurrence), we must compute the mean and standard deviation differently.

Let x and y denote the two quantities. We assume that x and y are independent. (This is valid since Size and Execution Rate are not coupled.) This means that:

$\langle xy\rangle = \langle x\rangle\langle y\rangle$

where $\langle x\rangle = (x_L + 4x_M + x_H)/6$

$\langle y\rangle = (y_L + 4y_M + y_H)/6$

The variance of the product xy is:

$$\text{variance}(xy) = ?_x^2 ?_y^2 + ?_x^2 \langle y\rangle^2 + ?_y^2 \langle x\rangle^2$$

where

$?_x = (x_H - x_L)/6$

$?_y = (y_H - y_L)/6$

The standard deviation of <xy> is the square root of variance (xy).